МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Искусственные нейронные сети»

Тема: «Классификация обзоров фильмов»

Студент гр. 7381 _____ Лукашев Р.С.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

Классификация последовательностей - это проблема прогнозирующего моделирования, когда у вас есть некоторая последовательность входных данных в пространстве или времени, и задача состоит в том, чтобы предсказать категорию для последовательности.

Проблема усложняется тем, что последовательности могут различаться по длине, состоять из очень большого словарного запаса входных символов и могут потребовать от модели изучения долгосрочного контекста или зависимостей между символами во входной последовательности.

В данной лабораторной работе также будет использоваться датасет IMDb, однако обучение будет проводиться с помощью рекуррентной нейронной сети.

Задачи.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Требования.

1. Найти набор оптимальных ИНС для классификации текста
2. Провести ансамблирование моделей
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
4. Провести тестирование сетей на своих текстах (привести в отчете)

Ход работы.

Были созданы и обучены две модели искусственной нейронной сети, решающей задачу определения настроения обзора. Первая модель является рекуррентно-свёрточной, а вторая – рекуррентной.

```
model1 = Sequential()
model1.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model1.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model1.add(MaxPooling1D(pool_size=2))
model1.add(Dropout(0.3))
model1.add(Conv1D(filters=64, kernel_size=3, padding='same', activation='relu'))
model1.add(MaxPooling1D(pool_size=2))
model1.add(Dropout(0.3))
model1.add(LSTM(100))
model1.add(Dropout(0.3))
model1.add(Dense(1, activation='sigmoid'))

model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рисунок 1 – архитектура первой модели.

```
model2 = Sequential()
model2.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model2.add(LSTM(100))
model2.add(Dropout(0.3))
model2.add(Dense(100, activation='relu'))
model2.add(Dropout(0.4))
model2.add(Dense(1, activation='sigmoid'))

model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рисунок 2 – архитектура второй модели.

Сети тренировались на трех эпохах, итоговая точность и потери каждой модели предоставлены на рисунках 3, 4.

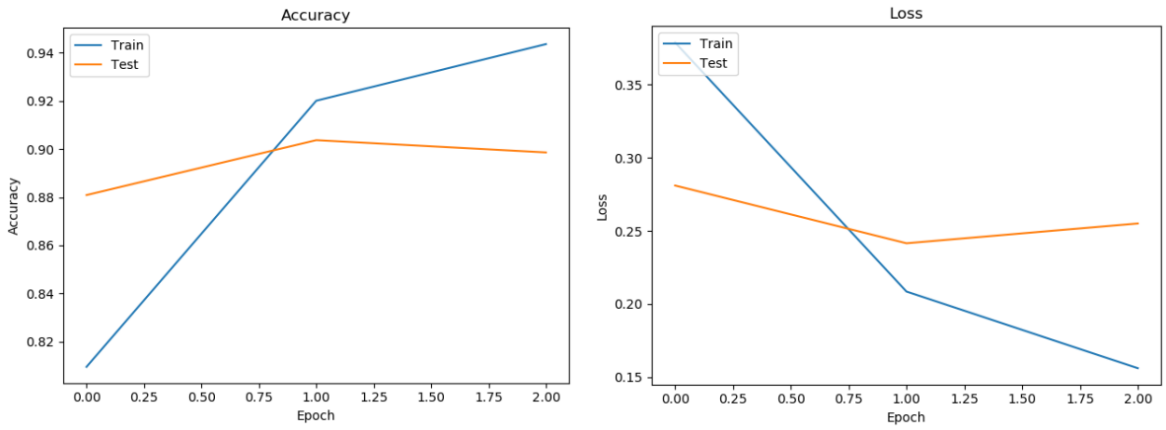

Рисунок 3 – точность и потери первой модели.

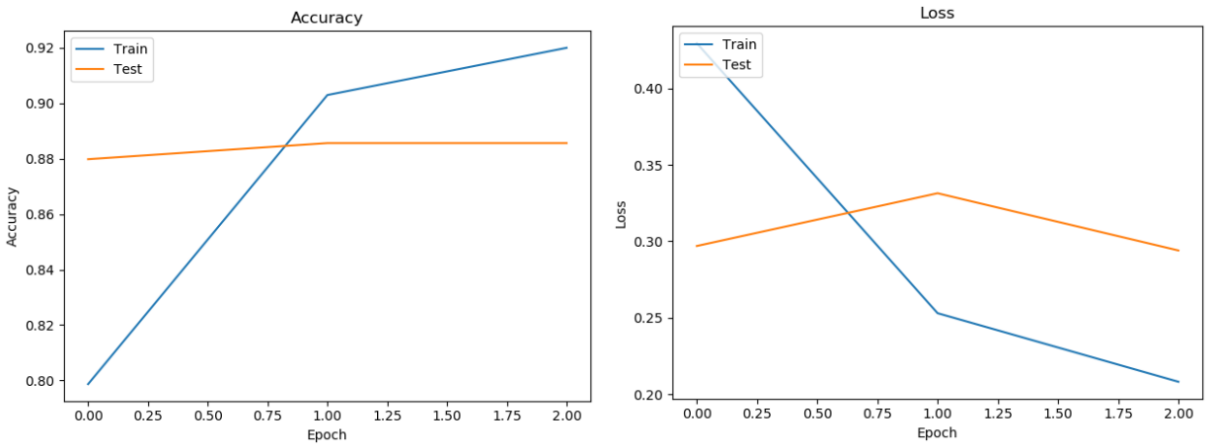

Рисунок 4 – точность и потери второй модели.

После было проведено ансамблирование сетей. Результатом работы ансамбля является среднее арифметическое предсказаний моделей. Точность предсказания ансамбля и результат его работы на пользовательском тексте “I'm not sure if I like this movie. Seems like a happy loss of time. Although the movie had a good cast and actor played actually very decently, the story is boring. Didn't really liked it.” Показаны на рисунке 5.

```
Ensemble accuracy: 0.8984
Ensemble predict: 0.34389398
```

Рисунок 5 – точность ансамбля на тестовой выборке и предсказание ансамбля на пользовательском тексте.

К сожалению, значительно повысить точность с использованием данным моделей с помощью выбранного метода ансамблирования так и не удалось, однако на пользовательском тексте ансамбль определил настроение обзора корректно.

Выводы.

В ходе выполнения лабораторной работы были созданы две модели нейронных сетей для предсказания успеха фильма по обзорам. Была проведена попытка улучшения точности предсказания с использованием ансамблирования сетей. Были написаны функции, с помощью которых было проведено тестирование модели на пользовательских данных.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import numpy as np
import re
import matplotlib.pyplot as plt
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, GRU, Conv1D, MaxPooling1D, Dropout
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
from keras.datasets import imdb


def plot_history(history):
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()


def build_models():
    models = []
    model1 = Sequential()
    model1.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    model1.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
    model1.add(MaxPooling1D(pool_size=2))
    model1.add(Dropout(0.3))
    model1.add(Conv1D(filters=64, kernel_size=3, padding='same', activation='relu'))
    model1.add(MaxPooling1D(pool_size=2))
    model1.add(Dropout(0.3))
    model1.add(LSTM(100))
    model1.add(Dropout(0.3))
    model1.add(Dense(1, activation='sigmoid'))

    model1.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

    model2 = Sequential()
    model2.add(Embedding(top_words, embedding_vector_length,
input_length=max_review_length))
    model2.add(LSTM(100))
    model2.add(Dropout(0.3))
    model2.add(Dense(100, activation='relu'))
    model2.add(Dropout(0.4))
    model2.add(Dense(1, activation='sigmoid'))

    model2.compile(loss='binary_crossentropy', optimizer='adam',
```

```
metrics=['accuracy'])

    models.append(model1)
    models.append(model2)

    return models


def train_models(train_x, train_y, test_x, test_y):
    models = build_models()
    model1 = models[0]
    model2 = models[1]

    plot_history(model1.fit(train_x, train_y, validation_data=(test_x, test_y),
epochs=epochs1, batch_size=batch_size1))
    print(model1.evaluate(test_x, test_y, verbose=0))
    model1.save('model1.h5')

    plot_history(model2.fit(train_x, train_y, validation_data=(test_x, test_y),
epochs=epochs2, batch_size=batch_size2))
    print(model2.evaluate(test_x, test_y, verbose=0))
    model2.save('model2.h5')
    return [model1, model2]


def test_ensemble():
    a = model1.predict(test_x)
    b = model2.predict(test_x)
    prediction = np.divide(np.add(a, b), 2)
    predictions = np.greater_equal(prediction, np.array([0.5]))
    targets = np.reshape(test_y, (np.size(predictions), 1))
    accuracy = np.mean(np.logical_not(np.logical_xor(predictions, targets)))
    ensemble_info = "Ensemble accuracy: " + str(accuracy)
    print(ensemble_info)


def ensemble(to_predict=None):
    if to_predict.all() is not None:
        return "Ensemble predict: " + str(np.mean((model1.predict(to_predict)[0][0],
model2.predict(to_predict)[0][0])))


def ensemble_predict(text):
    output_str = ''
    with open(text, 'r') as f:
        for input_str in f.readlines():
            output_str += re.sub('[^A-Za-z0-9 ]+', '', input_str).lower()
    indexes = imdb.get_word_index()
    encode = []
    text = output_str.split()
    for index in text:
        if index in indexes and indexes[index] < 10000:
            encode.append(indexes[index])
    encode = sequence.pad_sequences([np.array(encode)], maxlen=max_review_length)
    print(ensemble(encode))


embedding_vector_length = 32
top_words = 10000
max_review_length = 500
batch_size1 = 100
```

```
batch_size2 = 64
epochs1 = 3
epochs2 = 3

(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=top_words)

training_data = sequence.pad_sequences(training_data, maxlen=max_review_length)
testing_data = sequence.pad_sequences(testing_data, maxlen=max_review_length)

data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

test_x = data[:10000]
test_y = targets[:10000]
train_x = data[10000:]
train_y = targets[10000:]

[model1, model2] = train_models(train_x, train_y, test_x, test_y)
#model1 = load_model("model1.h5")
#model2 = load_model("model2.h5")
test_ensemble()
ensemble_predict('test.txt')
```